Road Safety Analysis

United Kingdom, 2020

Rareș Lițescu

Table of contents

About the project

- **What is the scope of the project?**

The project represents a short analysis of the **UK Road Safety Data** for the year 2020.

This document is meant to show the technical process behind the analysis, including all the code that was used along with the SQL queries and the thought process. As such, it is not meant for the general audience.

- **Which dataset was used?**

The dataset consists of three tables (*accident, vehicle* and *casualty*), which together show information about 91,199 accidents that happened in the United Kingdom during 2020.

More information about the data can be found at the following link:
https://www.data.gov.uk/dataset/cb7ae6f0-4be6-4935-9277-47e5ce24a11f/road-safety-data

The supporting documents which contain explanations for all the fields are included as well.

- **Which tools were used?**

The analysis was conducted using MySQL to fetch the data from the tables, which was then visualized using Tableau. Moreover, Python was also used along the way to make the process easier.

- **What about the style conventions of the document?**

To make it easier to distinguish between the different sections, they will have different formatting styles. The following are some examples from each:

> **Remark:** every remark will be formatted like this, using an orange background

```
# and python code will be formatted like this
format = 'black background'
```

```
-- sql queries will look like this
SELECT gray_background, black_borders
FROM format
```

Importing the files

We begin by checking the *.csv* files for their structure, taking note of the datatypes and possible connections between tables.

To import the files into the MySQL database, we first need to create the tables. Since each file has a large number of columns, we can write a custom Python script that goes through all the files and fields and outputs the SQL commands.

```
import pandas as pd

files = ['accident', 'vehicle', 'casualty']

output = str() #initiate an empty string

for file in files:
      df = pd.read_csv(f'{file}.csv',
                       low_memory=False)
    output += f'CREATE TABLE {file} (\n'
    for col in df.columns: #go through all columns
        #check the type of column
        if df.dtypes[col] == 'O':
            output += f'\t{col} VARCHAR(30),\n'
        elif df.dtypes[col] == 'int64':
            output += f'\t{col} INT,\n'
        elif df.dtypes[col] == 'float64':
            output += f'\t{col} DECIMAL(16, 8),\n'
    output = output[:-2] + '\n' #delete the comma at the end
    output += ');\n\n'

print(output)
```

Remark 1: the restrictions for the *string* and *float* fields (i.e. ***VARCHAR(30)*** and ***DECIMAL(16, 8)***) were chosen after inspecting the values in the *.csv* files

Remark 2: the output of the code above can be seen in the *appendix*[1].

After creating the tables, the *.csv* files need to be prepared for being imported into the database. This means changing empty cells to **\N**, which will be interpreted by MySQL as **NULL** values.

To do this, we create another Python script.

```python
import pandas as pd
import numpy as np

files = ['accident', 'vehicle', 'casualty']

for file in files:
    df = pd.read_csv(f'{file}.csv', low_memory=False)
    df.replace({np.NaN: '\\N'}, inplace=True)
    df.to_csv(f'{file}_prepped.csv', index=False)
```

Load the data into the database:

```sql
LOAD DATA INFILE 'E:/road_safety/accident_prepped.csv'
INTO TABLE accident
FIELDS TERMINATED BY ','
IGNORE 1 ROWS;

LOAD DATA INFILE 'E:/road_safety/vehicle_prepped.csv'
INTO TABLE vehicle
FIELDS TERMINATED BY ','
IGNORE 1 ROWS;

LOAD DATA INFILE 'E:/road_safety/casualty_prepped.csv'
INTO TABLE casualty
FIELDS TERMINATED BY ','
IGNORE 1 ROWS;
```

For the scope of this project, it is not necessary to worry about setting primary/foreign keys, since the database won't be updated in any way after loading in the files. Moreover, the fields that uniquely identify the records in each table have been checked in Python to ensure there are no duplicates or null values for those fields.

Analysis

Now that the database is all set up, it's time to finally start asking questions about the data.

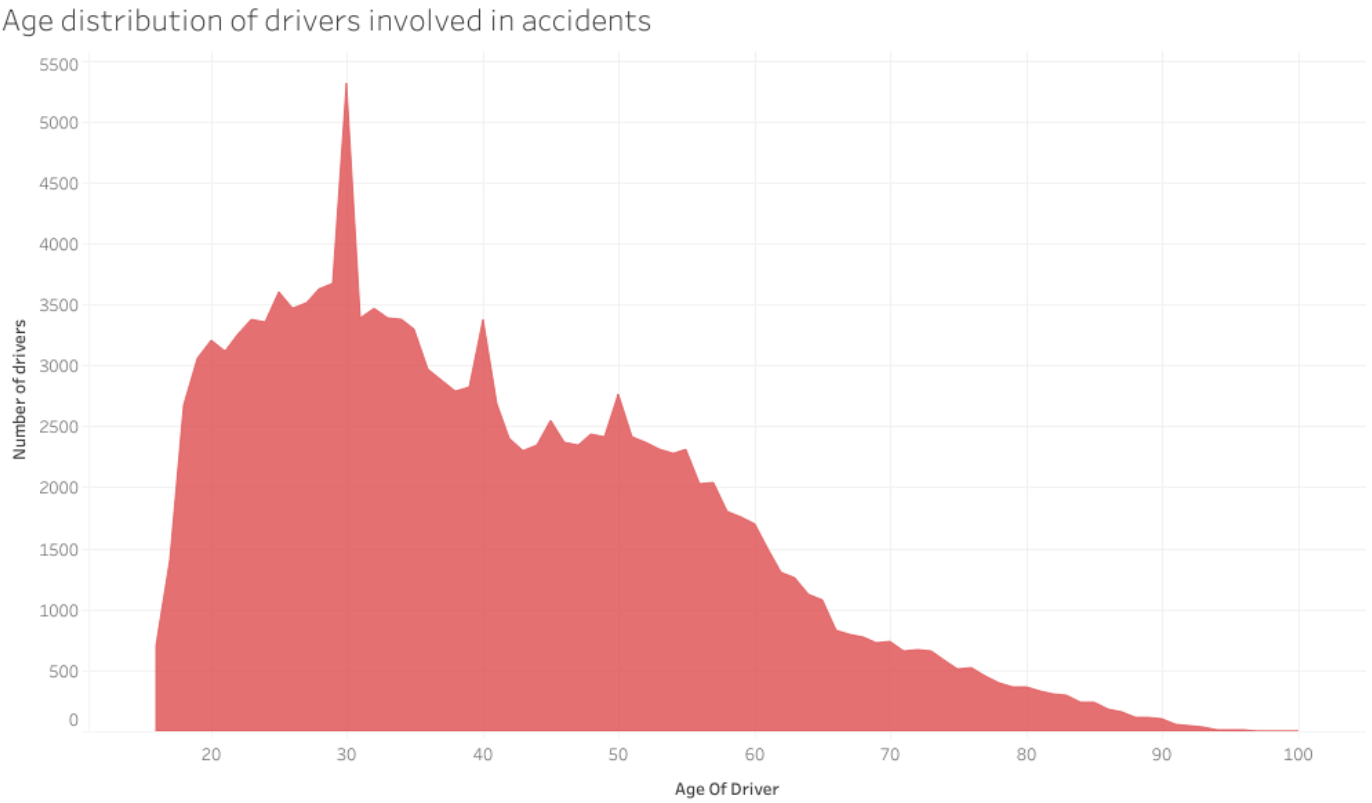
The result of each query will be exported as a *.csv* file and visualized using Tableau.

- **What is the age distribution of the drivers?**

To answer this question, we must look the *vehicle* table, which includes a field called *age_of_driver* that, as the name suggests, includes the ages of all the drivers involved in accidents.

Since there is no information about the accuracy of the measurements, ages lower than 16 will be ignored.

```
SELECT age_of_driver, COUNT(*) AS count
FROM vehicle
WHERE age_of_driver >= 16 # exclude invalid ages
GROUP BY age_of_driver
ORDER BY age_of_driver;
```



The average age of the drivers is 34.5 years.

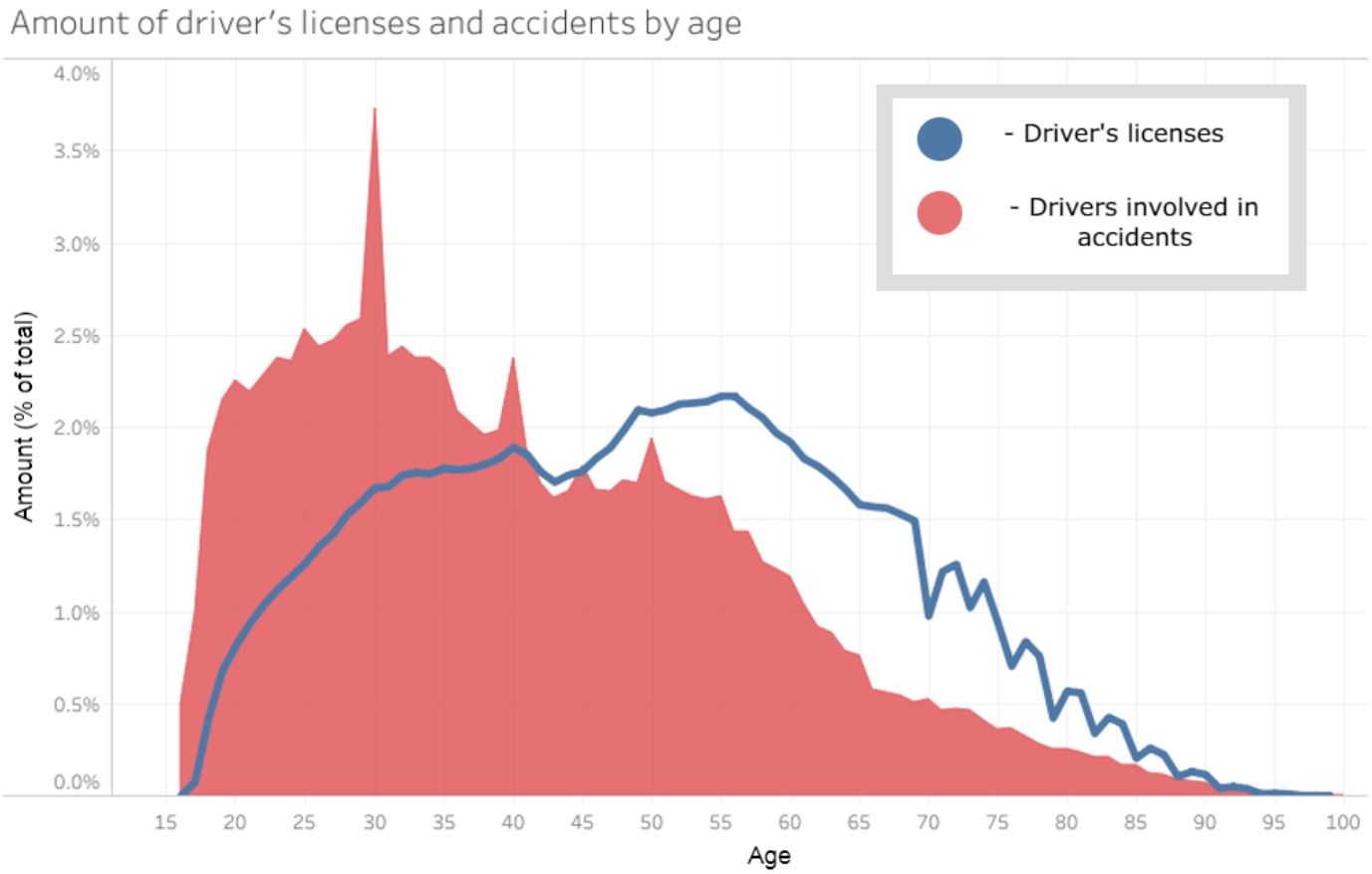
Remark: there are some spikes around multiples of 5 with bigger ones around multiples of 10. This might be due to approximation errors when determining the age of the people involved

- **How does this correlate with the driver's license distribution by age in the UK?**

To answer this, an additional dataset was used, which contains information about the amount of driver's licenses for each age.

It can be found at the following link: https://www.data.gov.uk/dataset/d0be1ed2-9907-4ec4-b552-c048f6aec16a/gb-driving-licence-data

The graph below shows the distributions for the drivers involved in accidents along with that for the number of licensed drivers by age. Both distributions are rescaled to percentages out of the total.



Remark: the legend was added later in Photoshop

Suppose the following two things hold true:

1. The risk of getting involved in an accident is the same for each age group
2. The frequency of car rides stays constant for each age group (younger drivers use their cars just as often as older people)

Then, we would expect the blue line to follow the red shape (meaning that the two distributions would be identical as percentages). However, the graph suggests otherwise, meaning that at least one of the assumptions is wrong.

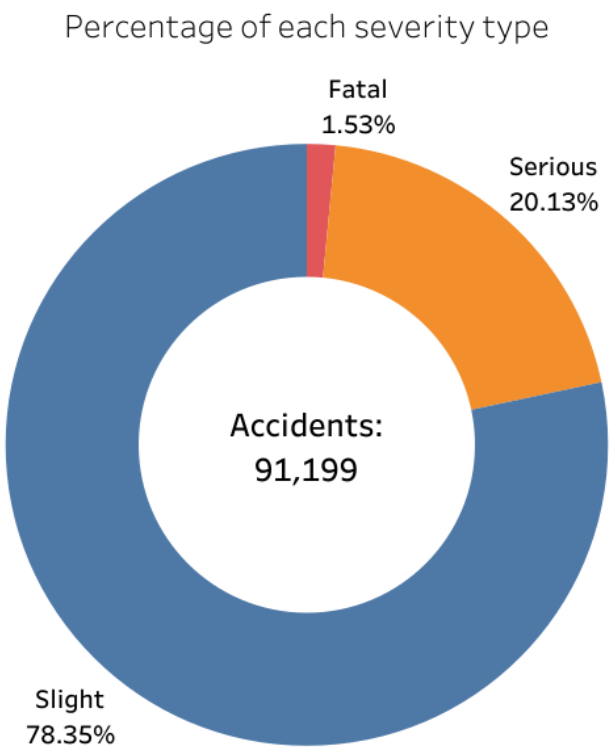
Younger people (under 35) are responsible for a larger share of accidents than older people (over 50) relative to the number of driver's licenses owned by each age group.

Possible explanations:

1. younger people engage more often in risky driving practices than older people
2. older people drive less often than younger people despite owning a driver's license
3. a combination of both

- **What about accident severity?**

```sql
SELECT accident_severity, COUNT(*) AS count
FROM accident
GROUP BY accident_severity
ORDER BY accident_severity DESC;
```



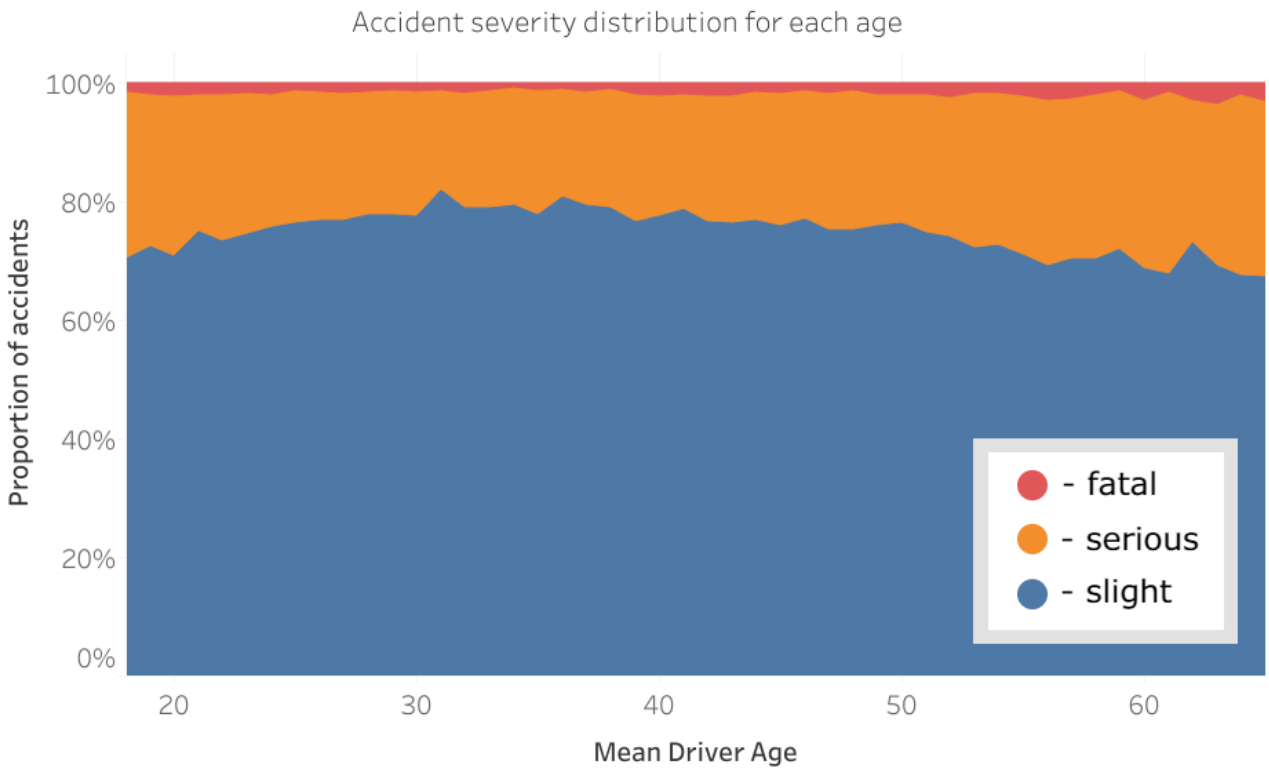
As shown in the graph above, the vast majority of the accidents (78.35%) are only *slight.* Meanwhile, only 1.53% of the accidents are *fatal*.

Since severity is a very important metric to keep track of, it can be analyzed further by visualizing its relationship with driver's age and car's age.

- **How is the age of the drivers related to accident severity?**

To show how the age of the drivers affects accident severity, there needs to be one single age-related value for each unique accident. Since there are on average 1.8 drivers involved in each accident (details in the *appendix*[2]), the we can consider the average age of the drivers involved in each accident.

```sql
SELECT
    a.accident_reference,
    FLOOR(AVG(age_of_driver)) AS mean_driver_age,
    MAX(accident_severity) AS accident_severity # could have chosen MIN as well
FROM accident a
JOIN vehicle v
ON a.accident_reference = v.accident_reference
# filter out accidents where some ages ae invalid
WHERE
    a.accident_reference IN
    (SELECT DISTINCT accident_reference
    FROM
        (SELECT
            accident_reference,
            MIN(age_of_driver) OVER(partition by accident_reference) AS min_age
        FROM vehicle) t
        WHERE min_age >= 16) # only choose accidents where the minimum driver age >= 16
GROUP BY a.accident_reference;
```



Remark: the age axis was restricted to values between 18 and 65 to make sure there are enough records (>400) for each age and thus ensure some level of statistical significance

The graph shows that the age of the drivers has a small influence on the severity of the accidents, with younger drivers (<23) and older drivers (>50) having a slightly higher chance of getting into serious and fatal accidents than middle aged drivers.

The statistical significance of the differences between the severity distribution by age can be checked using a chi-squared hypothesis test, with H_0: the distributions are the same and H_1: the distributions are different. This is used to verify that the differences are not due to random chance.

It can be implemented into Python, using the *.csv* file that was generated from the previous SQL query.

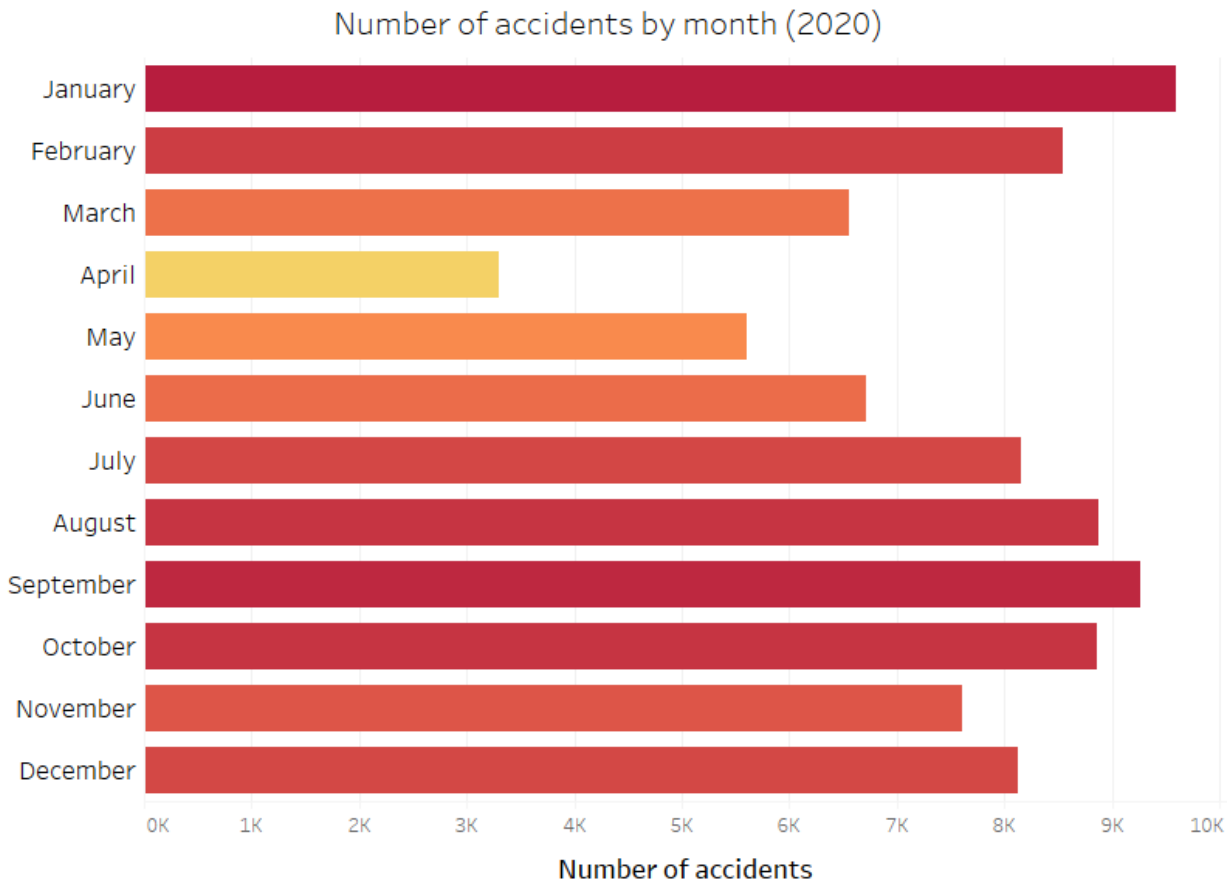
```
import pandas as pd
import numpy as np
from scipy.stats import chi2_contingency

severity_by_age = pd.read_csv('severity by driver age.csv')

#count of slight accidents by age
filt_slight = (severity_by_age['accident_severity'] == 3)
slight = severity_by_age[filt_slight].groupby('mean_driver_age')
slight = slight['mean_driver_age'].count()
#count of serious accidents by age
filt_serious = (severity_by_age['accident_severity'] == 2)
serious = severity_by_age[filt_serious].groupby('mean_driver_age')
serious = serious['mean_driver_age'].count()
#count of fata accidents by age
filt_fatal = (severity_by_age['accident_severity'] == 1)
fatal = severity_by_age[filt_fatal].groupby('mean_driver_age')
fatal = fatal['mean_driver_age'].count()

min_age=20
max_age=65
#check for ages between 20 and 65
list_of_proportions = list(zip(slight[min_age:max_age],
                      serious[min_age:max_age], fatal[min_age:max_age]))
chi2_contingency(list_of_proportions)
```

By running the code above, we get a *p*-value < .001, which means there is enough evidence reject the null hypothesis and to suggest that there are differences between the distributions.

More information about the test here: https://en.wikipedia.org/wiki/Chi-squared_test

- **What is the number of accidents for each month of 2020?**

```sql
SELECT
    MONTH(STR_TO_DATE(date, '%d/%m/%y')) AS month_number,
    MONTHNAME(STR_TO_DATE(date, '%d/%m/%y')) AS month,
    COUNT(*) AS count
FROM accident
GROUP BY month
ORDER BY month_number;
```



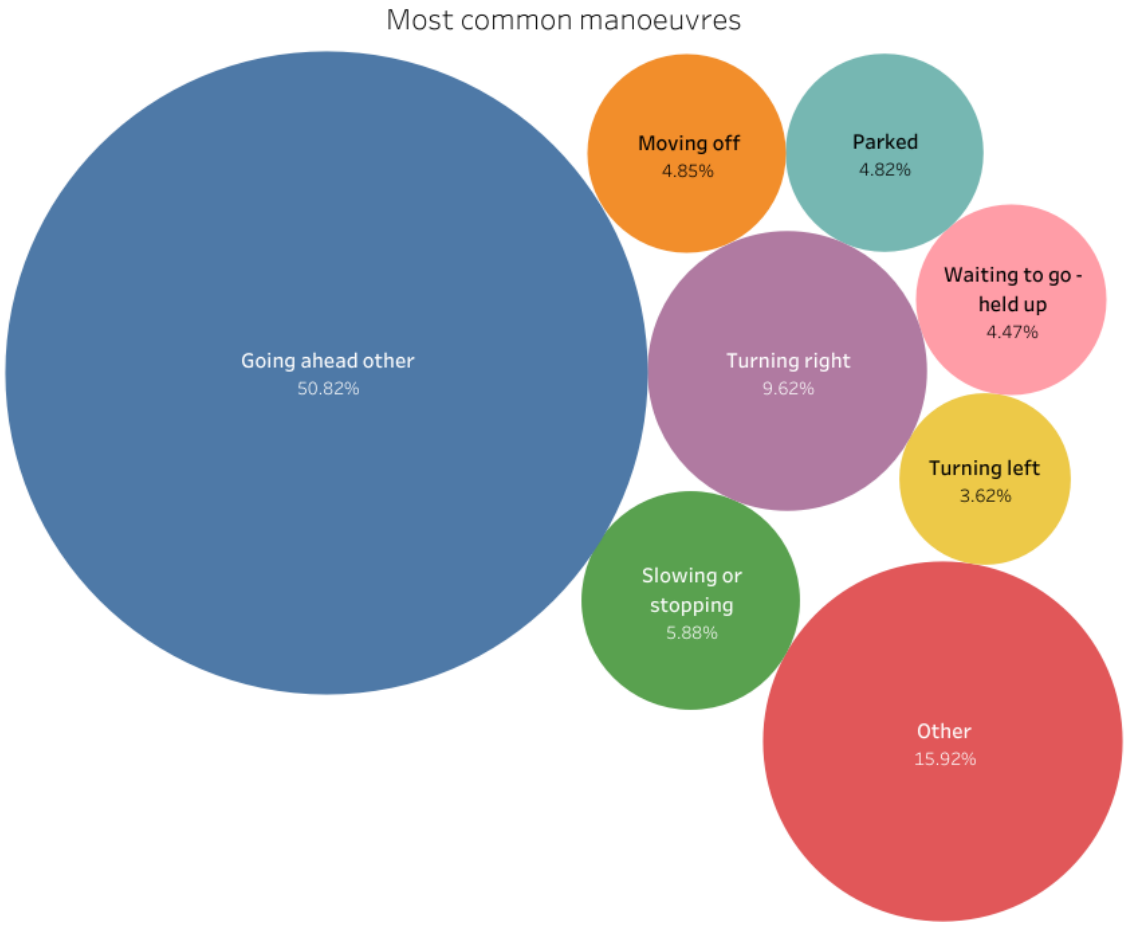
At first, the monthly number of accidents shown above might seem unexpected, having unusually low numbers around April.

However, notice that the dataset only covers accidents that happened during 2020, which was the year when the COVID-19 pandemic started. More specifically, the UK Government announced the first nationwide lockdown on the 23rd of March, 2020. This explains the uneven distribution of the accident frequency.

- **What are the most common vehicle manoeuvres in accidents?**

In total, there are 18 unique types of manoeuvres that are recorded in the dataset. When ranking them, only the 7 most common ones will be shown, to reduce clutter. The rest will be combined into the *other* category.

```sql
# most common manouvres in accidents

SELECT manoeuvre, SUM(count) AS count
FROM
    (SELECT
        CASE
            WHEN vehicle_manoeuvre = 18 THEN 'Going ahead other'
            WHEN vehicle_manoeuvre = 9 THEN 'Turning right'
            WHEN vehicle_manoeuvre = 4 THEN 'Slowing or stopping'
            WHEN vehicle_manoeuvre = 5 THEN 'Moving off'
            WHEN vehicle_manoeuvre = 2 THEN 'Parked'
            WHEN vehicle_manoeuvre = 3 THEN 'Waiting to go - held up'
            WHEN vehicle_manoeuvre = 7 THEN 'Turning left'
            ELSE 'Other'
        END AS manoeuvre,
        COUNT(*) AS count
    FROM vehicle
    WHERE vehicle_manoeuvre BETWEEN 1 and 18
    GROUP BY vehicle_manoeuvre
    ORDER BY count DESC) t
GROUP BY manoeuvre;
```



Remark 1: a very common manoeuvre with a low risk could result in the same number of accidents as an uncommon manoeuvre with a very high risk, so there is no way to estimate how risky each of them is based on the available data alone.

Remark 2: the values in the CASE WHEN clause were chosen after inspecting which manoeuvres are the most common

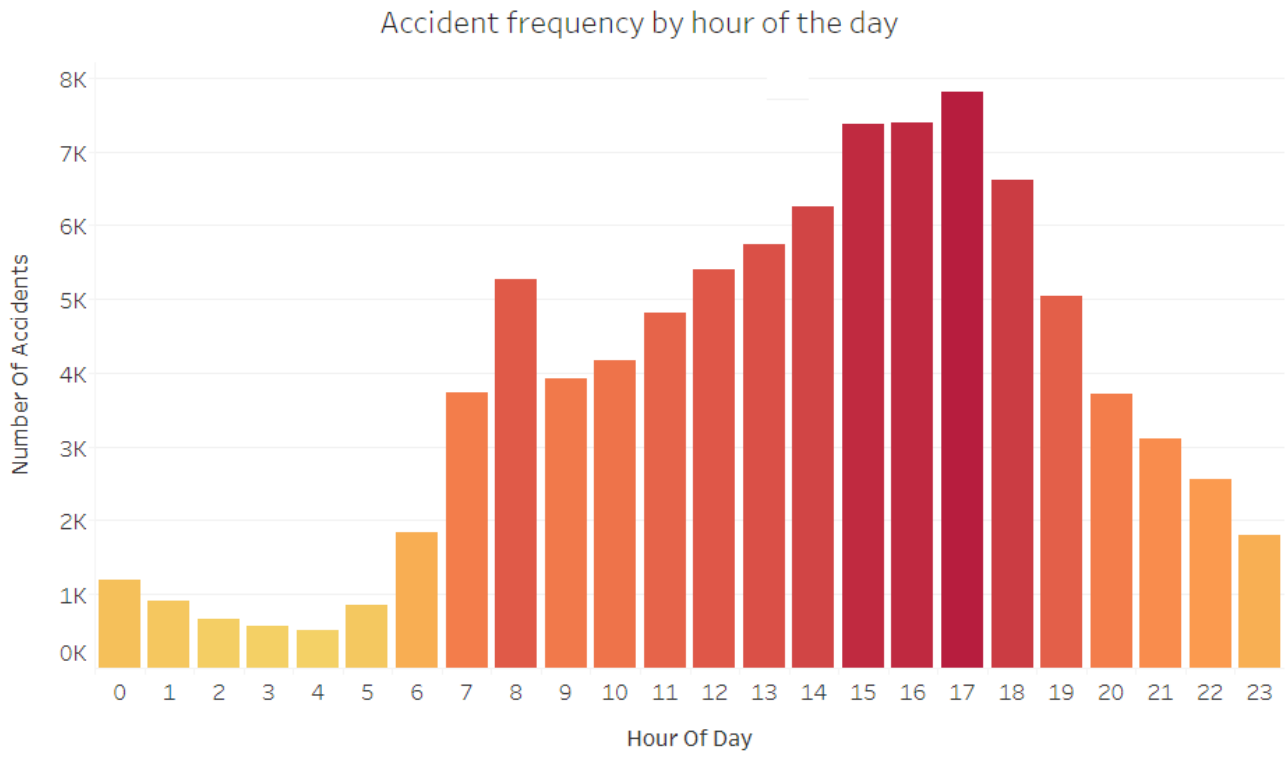
It's interesting to note that *turning right* appears almost 3 times more often than *turning left* in the dataset. Assuming that people turn in either direction just as frequently, it could mean that going right is almost 3 times more dangerous.

This could happen because turning right means crossing one or more extra lanes, which increases the risk of collisions.

In fact, this this hypothesis would mean that in the countries where people drive on the right, the opposite holds true.

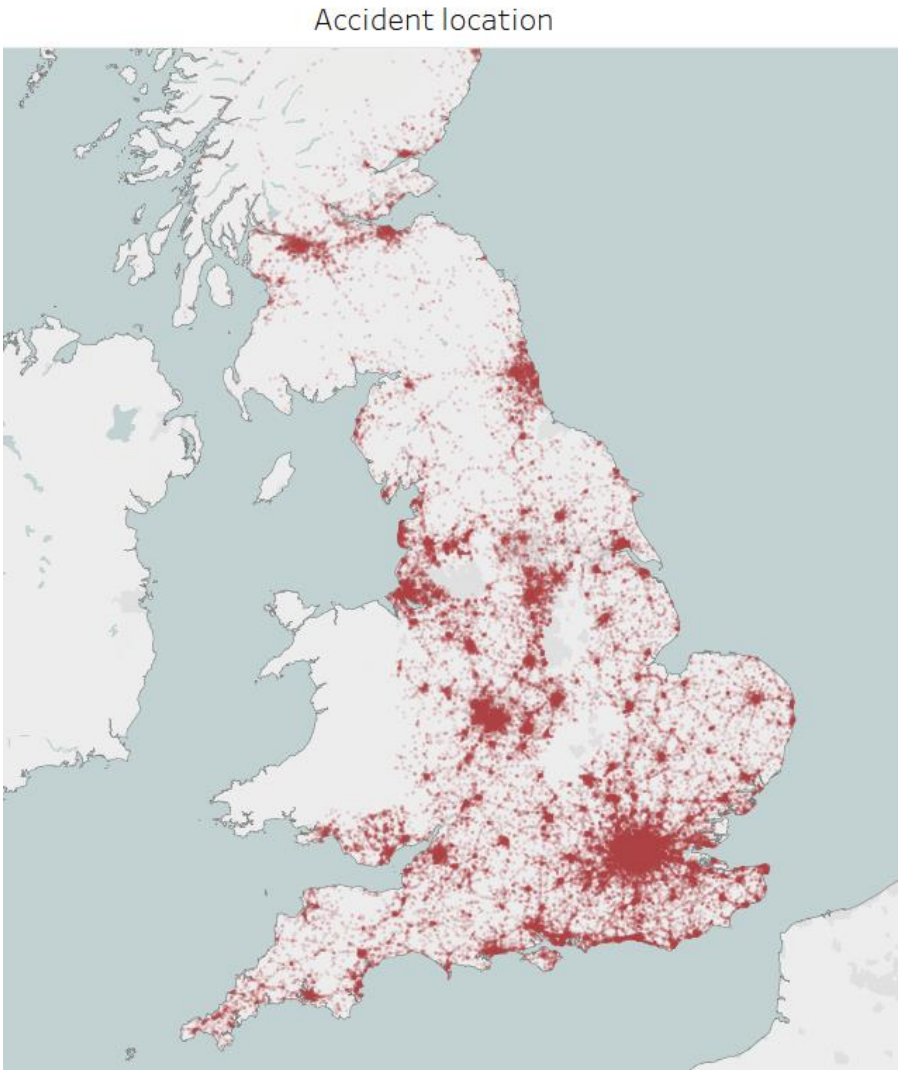
- **What is the frequency for accidents by hour of the day?**

```
SELECT
    hour(time) AS hour_of_day, COUNT(time) AS number_of_accidents
FROM accident
GROUP BY hour_of_day
ORDER BY hour_of_day;
```



As seen in the table above, most accidents happen around 17:00. Notice there is a spike around 8:00, which could be caused by people driving to work around that time.

- **What is the location of the accidents?**



To answer this question, we will import the *latitude* and *longitude* columns directly into Tableau.

Each red dot represents one accident.

To create the heatmap effect, the opacity and size of each dot were reduced so that only the areas with a high density of accidents show up.

As expected, most accidents happen in regions with high population density, such as cities and towns.

- **How does the lighting conditions affect severity?**

To visualize this, we need to come up with a formula for what will be called *severity score* of each lighting condition.

The *severity score* will be a numerical value that is directly proportional to the number of accidents in each category, where more severe accidents have a greater impact on the final score. To do this, each severity type will have a weight attached to it.

The weights are somewhat arbitrary, but they should reflect the gravity of each accident.

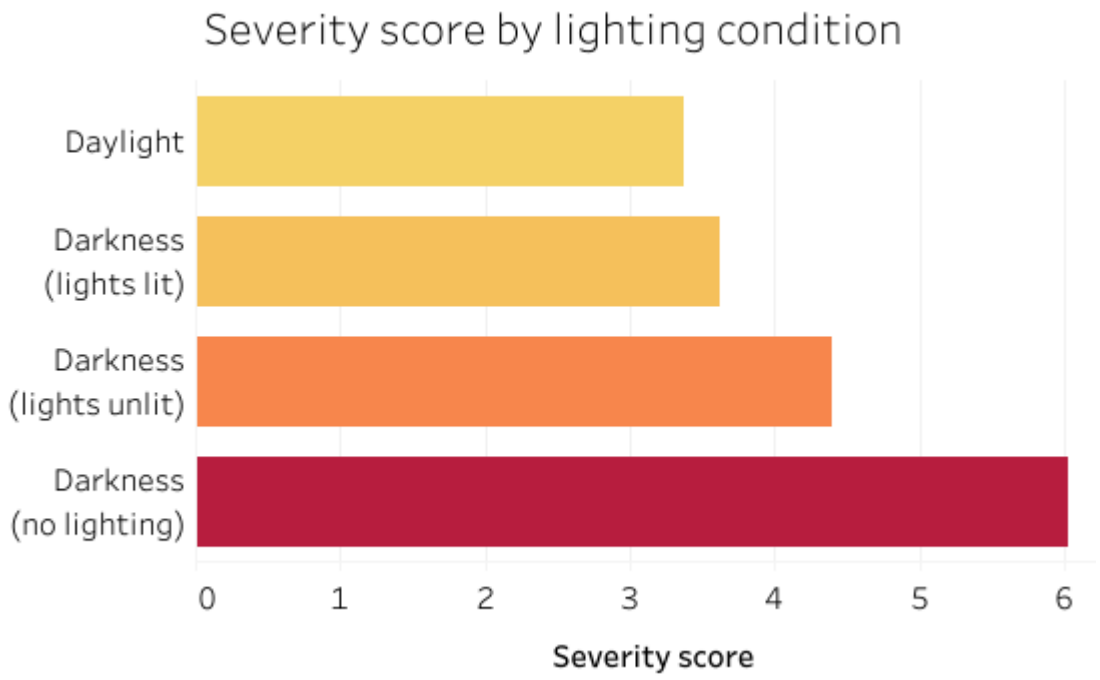
For our purpose, we will assign a base weight of 1 to slight accidents, a weight of 10 to serious accidents and a weight of 50 for fatal accidents.

Therefore, the formula for the severity score will be:

$$score = \frac{1 \cdot (\#\ slight\ accidents) + 10 \cdot (\#\ serious\ accidents) + 50 \cdot (\#\ fatal\ accidents)}{total\ number\ of\ accidents}$$

> **Remark:** the # symbol should be read as '*number of*'

```sql
SELECT
    CASE
        WHEN light_conditions = 1 THEN 'Daylight'
        WHEN light_conditions = 4 THEN 'Darkness (lights lit)'
        WHEN light_conditions = 5 THEN 'Darkness (lights unlit_'
        WHEN light_conditions = 6 THEN 'Darkness (no lighting)'
    END as lighting,
    # compute a severity score by giving weights to each severity type
    AVG(CASE
            WHEN accident_severity = 3 THEN 1 # slight accidents
            WHEN accident_severity = 2 THEN 10 # serious accidents
            WHEN accident_severity = 1 THEN 50 # fatal accidents
        END) AS severity,
    COUNT(*) AS number_of_accidents # make sure there are enough samples
FROM accident
WHERE light_conditions IN (1, 4, 5, 6) # choose the relevant categories
GROUP BY lighting;
```



> **Remark:** the absolute values of the severity scores have no meaning and should only be used as a relative measure

The graph above suggests that when accidents happen in poor lighting conditions, they tend to be more severe.

This can be seen from the severity distribution of each type of lighting conditions.

```
SELECT
    CASE
        WHEN light_conditions = 1 THEN 'Daylight'
        WHEN light_conditions = 4 THEN 'Darkness (lights lit)'
        WHEN light_conditions = 5 THEN 'Darkness (lights unlit)'
        WHEN light_conditions = 6 THEN 'Darkness (no lighting)'
    END AS light_condition,
    100 * slight/(slight + serious + fatal) AS slight_percentage,
    100 * serious/(slight + serious + fatal) AS serious_percentage,
    100 * fatal/(slight + serious + fatal) AS fatal_percentage
FROM(SELECT
    light_conditions,
    COUNT(CASE
        WHEN accident_severity = 3 THEN accident_severity ELSE NULL
    END) AS slight,
    COUNT(CASE
        WHEN accident_severity = 2 THEN accident_severity ELSE NULL
    END) AS serious,
    COUNT(CASE
        WHEN accident_severity = 1 THEN accident_severity ELSE NULL
    END) AS fatal
FROM accident
WHERE light_conditions in (1, 4, 5, 6)
GROUP BY light_conditions) temp;
```

The SQL query above results in the following table (*formatted for clarity*):

Lighting condition	Slight (%)	Serious (%)	Fatal (%)
Daylight	79.1	19.6	1.3
Darkness (lights lit)	77.7	20.8	1.5
Darkness (lights unlit)	75.2	21.9	2.9
Darkness (no lighting)	67.4	27.4	5.2

As shown in the table, the worse the lighting conditions are, the higher the percentage of serious and fatal accidents.

Appendix

1. Output of the Python code (creating the tables):

```
CREATE TABLE accident (
        accident_index VARCHAR(30),
        accident_year INT,
        accident_reference VARCHAR(30),
        location_easting_osgr DECIMAL(16, 8),
        location_northing_osgr DECIMAL(16, 8),
        longitude DECIMAL(16, 8),
        latitude DECIMAL(16, 8),
        police_force INT,
        accident_severity INT,
        number_of_vehicles INT,
        number_of_casualties INT,
        date VARCHAR(30),
        day_of_week INT,
        time VARCHAR(30),
        local_authority_district INT,
        local_authority_ons_district VARCHAR(30),
        local_authority_highway VARCHAR(30),
        first_road_class INT,
        first_road_number INT,
        road_type INT,
        speed_limit INT,
        junction_detail INT,
        junction_control INT,
        second_road_class INT,
        second_road_number INT,
        pedestrian_crossing_human_control INT,
        pedestrian_crossing_physical_facilities INT,
        light_conditions INT,
        weather_conditions INT,
        road_surface_conditions INT,
        special_conditions_at_site INT,
        carriageway_hazards INT,
        urban_or_rural_area INT,
        did_police_officer_attend_scene_of_accident INT,
        trunk_road_flag INT,
        lsoa_of_accident_location VARCHAR(30)
);
```

```
CREATE TABLE vehicle (
        accident_index VARCHAR(30),
        accident_year INT,
        accident_reference VARCHAR(30),
        vehicle_reference INT,
        vehicle_type INT,
        towing_and_articulation INT,
        vehicle_manoeuvre INT,
        vehicle_direction_from INT,
        vehicle_direction_to INT,
        vehicle_location_restricted_lane INT,
        junction_location INT,
        skidding_and_overturning INT,
        hit_object_in_carriageway INT,
        vehicle_leaving_carriageway INT,
        hit_object_off_carriageway INT,
        first_point_of_impact INT,
        vehicle_left_hand_drive INT,
        journey_purpose_of_driver INT,
        sex_of_driver INT,
        age_of_driver INT,
        age_band_of_driver INT,
        engine_capacity_cc INT,
        propulsion_code INT,
        age_of_vehicle INT,
        generic_make_model VARCHAR(30),
        driver_imd_decile INT,
        driver_home_area_type INT
);

CREATE TABLE casualty (
        accident_index VARCHAR(30),
        accident_year INT,
        accident_reference VARCHAR(30),
        vehicle_reference INT,
        casualty_reference INT,
        casualty_class INT,
        sex_of_casualty INT,
        age_of_casualty INT,
        age_band_of_casualty INT,
        casualty_severity INT,
        pedestrian_location INT,
        pedestrian_movement INT,
        car_passenger INT,
```

```
        bus_or_coach_passenger INT,
        pedestrian_road_maintenance_worker INT,
        casualty_type INT,
        casualty_home_area_type INT,
        casualty_imd_decile INT
);
```

2.

```
# average number of drivers per accident


SELECT
    AVG(count_of_drivers) AS average
FROM
    (SELECT
        accident_reference,
        MAX(vehicle_reference) AS count_of_drivers
    FROM vehicle
    GROUP BY accident_reference) t
```